IJCSE International Journal of Computer Sciences and Engineering **Open Access**

Research Paper **Vol.-6, Issue-9, Sep 2018** **E-ISSN: 2347-2693**

Object Oriented Coupling based Test Case Prioritization

Ajmer Singh[1*], Rajesh Kumar Bhatia[2], Anita Singhrova[3]

[1] CSE Department, DCRUST, Murthal, India
[2] CSE Department, PEC University, Chandigarh, India
[3] CSE Department, DCRUST,Murthal,India

[*]*Corresponding Author: ajmer.saini@gmail.com, Tel.: +91-9813687398*

Available online at: www.ijcseonline.org

Accepted: 24/Sept/2018, Published: 30/Sep/2018

***Abstract*—** Test case prioritization is the process of ordering the test case executions to meet various testing goals. Improved software quality is one of the ultimate goals of software testing process. While dealing with object oriented testing specific features like inheritance, polymorphism and abstraction play important role in test case prioritization. Measures of object oriented features in a software component can be used as quality indicators. Some of the object oriented features may contribute positively while some may contribute negatively to the software quality. Software components can be ranked on the basis of their contribution in overall quality of software. These rankings may be helpful in test case prioritization. This study presents a test case prioritization approach based on software quality aspect. We performed an empirical investigation on six sequential versions of open source software and analyzed the contribution of various object oriented metrics in quality of the software. The work presents a novel technique of test case prioritization that would not only enhance the quality of the software but also prioritize the test cases as per fault proneness of the software modules. Proposed approach first investigates the impact of coupling metrics on software quality then provides ranking to component classes as per coupling measures. It is observed that coupling metrics potentially correlate with the change impact of software.

***Keywords*—** Regression Testing,Test case Prioritization, Machine Learning,Object Oriented metrics, Software Quality, Faults Prediction, Object Oriented Testing.

I. INTRODUCTION

Test case prioritizing is to give order to test cases in test suit to meet various testing goals. In case of model base testing the behaviour of software under test (SUT) is depicted by some sort of model. The degree of interdependence of a class to other class is described as coupling [1]. It is a well established fact that coupling may affect the change dependence in software. Tighter the coupling is, higher this dependence would be. Coupling can be of two types: static and dynamic. To estimate the dynamic coupling of software, run time analysis of the software is required. But static coupling can be estimated by examining the dependency model of SUT. Also it has been empirically established that coupling measures of SUT can be used as indicators of ripple effect and change impact. The software components with high measure of coupling have higher probability of containing the ripple effect [2]. There are empirical evidences that coupling does affect the structural complexity of software [3]. And coupling effect manifest itself emphatically when restricted to mutation analysis [4]. Thus when software is modified as a part of maintenance activity, it is expected that a change in highly coupled component would have higher possibility of stimulated faults as compared to a change in loosely coupled component. In case of regression testing, testers generally have the knowledge of test case coverage. The prior coverage information accompanied with coupling measures can be used to prioritize test cases in the subsequent version(s) of the software. The underlying hypothesis is that object oriented coupling has impact on the software quality. And a low quality module is more prone to faults as compared to high quality module.

Based on this hypothesis, a novel technique of test case prioritization is presented in this research work. This work is organized into five sections as follows: Section II describes the related work in filed. Section III explains proposed methodology for our study. Section IV outlines the empirical evaluation. Section V presents conclusions and future directions.

II. RELATED WORK

Test case prioritization has been a vast area of research in recent years. Diverse studies exist in literature in this domain. But most of the research work done previously deals with procedural programming concepts. So far, a very little work has considered the impact of object oriented features on test case prioritization. This section reviews some of the studies that have investigated the problem of prioritization in context of object oriented testing. It is found that most of the object oriented prioritization techniques have used model based testing. Model based testing is a technique to generate test cases on the basis of requirements [5]. Prioritization of test cases in model based testing uses different models like Activity diagrams, Finite state Machines (FSM) models, Extended FSM, UML state diagrams, Test Dependency Graphs (TDG), Object oriented System Dependency Graphs (OSDG) and Event Sequence Graphs. Research work of [6, 7, and 8] achieved the prioritization by analysing the changes in states of system models with the execution of test cases. Panigrahi and Mall [9] used extended - OSDG to perform test case prioritization for object oriented programs and they used backward and forward slicing techniques to identify the model elements that might have been affected by a change in software. Highest priority was assigned to test case that covered maximum number of such elements. Research work of Korel and Koutsogiannakis [10] experimentally compared the performance of coverage based TCP and model based TCP. Results of [10] observed significant improvement in early fault detection by model based TCP. Gantait [11] used extended UML activity diagrams to generate an intermediate model and then generated test cases on the basis of target test environment and prioritized them on the basis of coverage of transitions in activity model. Kundu et al [12] proposed system level test case generation and prioritization technique for object oriented testing named System Testing for Object-Oriented systems with test case Prioritization (*STOOP*). STOOP works on criteria of scenario coverage used sequence graphs (SG) models. Jaroenpiboonkit and Suwannasart [13] used an approach to find test case order by using Test Dependency Graph (TDG). TDG is used to represent relationship among the classes of SUT. Object oriented slicing is used to break the cyclic dependency of classes and to find the order of testing. Acharya and Mahali [14] used UML activity diagrams to model the SUT which was later converted into activity graphs. Authors applied association rules from the data mining to get frequently affected nodes in the model. Vedpal, Chauhan and Kumar [15] proposed hierarchical approach for prioritizing test cases of object oriented software. Authors used two level criteria for prioritization of test cases. In first level classes used in software, are prioritized on the basis of inheritance information and in second level test cases are ordered using inheritance weights of the classes covered. Authors in [44] presents the review of various object oriented coupling based test case selection techniques.

III. PROPOSED METHODOLOGY

This section highlights the various components of the proposed methodology.

A. *Outline of proposed methodology*

In regression testing modules covered by a test case can easily be traced from the prior runs of that test case. The modules covered can be analyzed for different OO metrics. Out of which, some metrics that are related with software coupling can be identified. Each module can be assessed for these coupling oriented measures. And coupling based weights can be assigned to these modules. From all the modules a test case covers; aggregate weight for that test case can be computed. Based upon aggregate weights, test cases can be provided corresponding ranks. The figure 1 below gives the outline of the proposed methodology

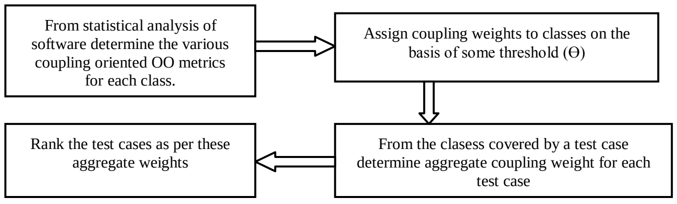
From statistical analysis of software determine the various coupling oriented OO metrics for each class.
Assign coupling weights to classes on the basis of some threshold (Θ)
From the clasess covered by a test case determine aggregate coupling weight for each test case
Rank the test cases as per these aggregate weights

Figure. 1. Outline of the proposed methodology

B. *Object Oriented Metrics*

Coupling is a measure of inter-dependencies among the software components. Features like inheritance, encapsulation and polymorphism manifest the interdependency object oriented software [16]. Information of coupling can be used to predict fault proneness and other costs of software [17]. The metrics like Chidamber and Kemerer object-oriented metrics suite (CK) [18], Tang & Chen metrics [19], Henderson Metrics [20], Li and Henry Metrics [21], Li Metrics [22], MOOD metrics [23] and QMOOD metrics [24] have been used in literature to assess various dependencies in software. Out of these, CK metrics have most extensively been used for predicting maintainability, fault proneness and quality of software [25]. In this research we look forward to analyze relationship between CK metrics and the quality of software modules. Software modules are weighted as per their contribution in software quality.

C. *Measuring Coupling in OO software*

In object oriented software, coupling is not only contributed by control, data, stamp, hybrid, content and common types but also by other types like coupling between objects (CBO), Depth of inheritance tree (DIT) and Number of Children (NOC). There are different measures of coupling in object oriented software like: structural coupling measure, dynamic coupling measure, logical coupling measure [26]. Where the structural coupling represents the inter connection of various structural components like classes, objects and method. Structural coupling does not require running the software for its measurement. Other types of coupling can be interpreted accordingly. Since structural complexity can be estimated from models like object relational diagrams (ORD), test sequence diagrams (TSG), unified modelling language

(UML) diagrams and dependency diagrams. This study utilises dependency graphs to estimate the coupling measure. A unit of testing in object oriented scenario can be either a class or a method. Similarly the coupling can be method level coupling and class level coupling. A class level coupling is evaluated and analyzed in this work. Following section gives the algorithm to determine coupling weight for a component class.

D. Proposed algorithm to determine Coupling weight

```
Input: Regression Test Suite T
Output: Accumulated object oriented coupling weight for every Test case is computed
Step 1. Form a class dependency graph G= (V, E) where V is set of nodes and E set of edges.
2. for every test case $Ti \in T$ compute a set of nodes $Si \subseteq G$ which represents the nodes covered by $Ti$.
3. For every node $Nj \in Si$ compute $OOc(Nj) = \sum_{k=1}^{t} (Ck)$
   Where t is number of various elements of node $Nj.$ and $OOc(Nj)$ is total coupling contribution of all t elements present at node $Nj.$
4. Compute the accumulative coupling weight covered by each test case $Ti \in T$ by
   $Wcc(Ti) = \sum OOc(Nj) : Nj \in Si$
5. Output $Wcc(Ti)$ for every $Ti \in T$
```

Figure 2. Proposed Algorithm to determine coupling weight.

The fig. 2 shown above depicts the steps involved in determining the coupling weights of the testing modules.

E. Assigning weight to software components

Every software module has its own measure of coupling. This measure indicates module's significance in fault prediction and quality assessment. Coupling weights can be assigned on the basis of derived impact of coupling of current module on the quality. In order to validate the weight assignment strategy, the correlation and regression analysis of coupling with the quality of the modules is performed. Also some acceptable threshold (Θ) of coupling is required to compare the coupling of different components and to provide relative weights. Difference of coupling of current module and Θ would indicate how far the module is from acceptable module. And the module that are at greatest difference are provide that have most negative coupling impact on quality.

F. Quality Prediction Model

Software quality is a multivariate problem and software aspects like inheritance, coupling of objects, cohesion, size, cyclometic complexity etc contribute to it. Low software quality is one of the derivatives of fault proneness of a software component. Object oriented metric like CK, MOOD, QMOOD, and Martin's etc have been used to assess the complexity and fault proneness of the software. Basili, Briand, and Melo [27] ,Briand et al. [28], El Emam, Melo, and Machado[29], Gyimothy, Ferenc, and Siket[30], Zhou and Leung [31], Catal, and Diri [32], Alan and Catal [33], Singh, Kaur, and Malhotra [34], Malhotra and Bansal [35], Xu, Ho, and Capretz [36] are some of the studies that have investigated effectiveness of CK metrics in fault prediction. The studies advocate that CK metrics are the indicators of the fault proneness of software modules. The prediction may also be made on other metrics like MOOD, QMOOD and Martin's metrics. There are empirical evidences that performance of QMOOD and CK metrics are almost similar [37]. Compositions of these metrics may also be incorporated to associate fault proneness with these software ingredients. In some of studies composition of Martins and MOOD performed better than the composition of CK and MOOD [38]. Also the metrics have been investigated for fault localization [39].But usage of CK metrics suite has been most as compared to others. We adopted a multivariate regression model to predict the quality of software modules in SUT. The model is depicted in brief in next section.

G. Multiple Linear Regression Model

Multi-linear regression is the approach of statistical data modelling, when the dependent variable and independent variables are linearly dependent. And the dependent variable is dependent on more than one independent variable. Formally it is described as under

$$\mathbf{Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k + \varepsilon} \quad (1)$$

Where Y is a dependent variable and $\mathbf{X_1, X_2 \ldots X_k}$ are the independent variables**,** $\mathbf{\beta_0, \beta_1 \ldots \beta_k}$, are the regression coefficients and **ε** is random error variable. The random error variable ε has zero mean.

H. Independent Variables for study

The various measures of a class such as WMC, DIT, NOC, DIT, RFC, CBO and LCOM control the quality of it. Some of these may be favourable for the quality while some may affect it negatively. Software with low coupling and high cohesion is considered to be more reliable, manageable and less prone to faults. To analyze the impact of various metrics on the quality of software, we initially considered all of the seven as independent variables for this study. All of these are static metrics and can be calculated with automated software like ckjm [40], JMT [41], and STAN [42]. Where ckm and JMT are open source and freeware. And STAN is proprietary software. JMT is used to extract these metrics for 6 different versions ANT software.

I. Dependent Variables

Quality of software is dependent on many intrinsic features of it. Low coupling is one of such essential features. To estimate the overall quality of software comprehensive information regarding testing, design and implementation is required. But individual class in software may be assessed for quality by analyzing some quantifiable features of it. JMT measures the quality of a software mode on the scale of 0 to 100. Where 0 represents the obvious least value of quality and 100 as best. We estimated the class level quality of the software version and considered it as dependent variable for analytical purpose.

It should be noted that low quality modules are difficult to maintain and are more prone to faults. So estimation of quality of module is inherently associated with fault proneness it.

J. Test vectors

Being dependent on multiple object oriented features the Object oriented complexity can be modelled multiple regression vector. The dimensions of the vector space are determined by various variable parameters contributing to OO complexity. And a test case that covers various OOPs components can also be presented as vector. We call it Test vector. It can be modelled like

$$T = a_1 x_1 + a_2 x_2 + a_3 x_3 + \cdots + a_n x_n \qquad (2)$$

Where $x_1, x_2 \ldots x_n$ be the multi – dimensional input variables indicting the OO metrics of the software and $a_1, a_2 \ldots a_n$ be their corresponding weights.

IV. EMPIRICAL EVALUATION

The independent variables of this model are PIM, WMC, WAC, NMI, MNO, DIT, NOC, CBO, and RFC. The Quality of the module is the dependent variable. The six successive version of Ant software i.e. Ant 1.6, Ant 1.6.1, 1.6.2, 1.6.3, 1.6.4 and 1.6.5 were statistically examined. The data about the various attributes is collected with the help of JMT tool. The tables 1 to 6 given below show the statistics of the different versions. It is observed that two successive versions of the software differ very little. So regression test suite can be applied to successive version(s). Also a mathematical regression model is derived for each version of the software. Quality of the module is a function of independent variables. The equations from 3 to 8 shown below are the regression models for six different versions. The coefficients in regression equation signify the correlation of any OO metric with quality. Also the sign of coefficients signify whether the correlation is positive or negative.

Table 1. Analysis of Version 1.6.0

Metric	Min	Max	Mean	Standard Deviation
PIM	0	121	13.597	17.099
WMC	0	100	8.878	10.677
WAC	0	45	4.237	6.018
NMI	0	115	11.631	18.592
MNO	0	30	0.706	1.774
DIT	2	8	3.098	1.245
NOC	0	59	0.522	3.228
CBO	2	50	11.183	8.722
RFC	0	272	31.817	36.778
QUALITY	40	100	84.68	14.208

Linear Regression Model-1

$$\text{Quality} = (0.0977)\text{PIM} + (-.6011)\text{WMC} + (-0.5031)\text{WAC} + (-0.3049)\text{NMI} + (-0.5049)\text{MNO} + (-1.7979)\text{DIT} + (-0.3743)\text{NOC} + (-0.6075)\text{CBO} + (0.0665)\text{RFC} + 105.1645 \qquad (3)$$

Correlation coefficient	0.9419
Mean absolute error	3.459
Root mean squared error	4.7855
Relative absolute error	29.4758 %
Root relative squared error	33.6705 %
Total Number of Instances	575

Table 2. Analysis of Version 1.6.1

Metric	Min	Max	Mean	Standard Deviation
PIM	0	121	13.506	16.764
WMC	0	100	8.79	10.684
WAC	0	45	4.114	5.861
NMI	0	107	11.543	18.19
MNO	0	30	0.696	1.812
DIT	2	8	3.108	1.238
NOC	0	58	0.574	3.442
CBO	2	51	11.103	8.504
RFC	0	273	30.538	34.113
QUALITY	40	100	84.716	13.817

Linear Regression Model-2

$$\text{QUALITY} = (0.0675)\ \text{PIM} + (-0.5686)\ \text{WMC} + (-0.4788)\ \text{WAC} + (-0.2831)\ \text{NMI} + (-0.4557)\ \text{MNO} + (-1.8249)\ \text{DIT} + (-0.3878)\ \text{NOC} + (-0.6216)\ \text{CBO} + (0.0658)\ \text{RFC} + 05.1419 \qquad (4)$$

Correlation coefficient	0.9376
Mean absolute error	3.5186
Root mean squared error	4.8206
Relative absolute error	31.0394 %
Root relative squared error	34.8752 %
Total Number of Instances	510

Table 3. Analysis of Version 1.6.2

Metric	Min	Max	Mean	Standard Deviation
PIM	0	121	13.424	16.852
WMC	0	102	8.835	10.824
WAC	0	45	4.185	6.067
NMI	0	112	11.332	18.239
MNO	0	30	0.654	1.78
DIT	2	8	3.077	1.228

NOC	0	59	0.552	3.371
CBO	2	53	10.491	8.337
RFC	0	273	30.538	34.113
QUALITY	35	100	84.908	14.198

Linear Regression Model-3

QUALITY = (0.0767) PIM +(-0.479) WMC + (-0.4999) WAC + (-0.2901) NMI +(-0.4832) MNO + (-1.922) DIT + (-0.3871) NOC + (-0.463) CBO + 104.7885 (5)

Correlation coefficient	0.9415
Mean absolute error	3.5076
Root mean squared error	4.7993
Relative absolute error	30.259 %
Root relative squared error	33.8008 %
Total Number of Instances	552

Table 4. Analysis of Version 1.6.3

Attribute	Min	Max	Mean	Standard Deviation
PIM	0	121	13.32	17
WMC	0	107	8.899	11.124
WAC	0	45	4.177	6.148
NMI	0	115	11.247	18.219
MNO	0	30	0.678	1.756
DIT	2	8	3.087	1.21
NOC	0	60	0.571	3.401
CBO	2	56	10.358	8.397
RFC	0	273	30.538	34.113
QUALITY	34	110	85.037	14.232

Linear Regression Model-4

QUALITY = (0.0597) PIM + (-0.4553) WMC + (-0.4829)WAC +(-0.2853) NMI + (-0.4733) MNO + (-1.8479) DIT + (-0.3889) NOC + (-0.4651) CBO + 104.5835 (6)

Correlation coefficient	0.9403
Mean absolute error	3.6117
Root mean squared error	4.8493
Relative absolute error	31.0185 %
Root relative squared error	33.9862 %
Total Number of Instances	575

Table 5. Analysis of Version 1.6.4

Attribute	Min	Max	Mean	Standard Deviation
PIM	0	121	13.323	17.001
WMC	0	107	8.903	11.126
WAC	0	45	4.179	6.149
NMI	0	115	11.247	18.219
MNO	0	30	0.678	1.756
DIT	2	8	3.087	1.21
NOC	0	60	0.571	3.401
CBO	2	56	10.36	8.4
RFC	0	273	30.538	34.113
QUALITY	34	100	85.03	14.237

Linear Regression Model-5

QUALITY = (0.0588) PIM + (-0.4548)WMC + (-0.4827) WAC + (-0.2846)NMI + (-0.4736) MNO + (-1.8471) DIT + (-0.3887) NOC + (-0.4655)CBO + 104.57 (7)

Correlation coefficient	0.9402
Mean absolute error	3.6153
Root mean squared error	4.8524
Relative absolute error	31.0457 %
Root relative squared error	33.9968 %
Total Number of Instances	575

Table 6. Analysis of Version 1.6.5

Attribute	Min	Max	Mean	Standard Deviation
PIM	0	121	13.325	17.003
WMC	0	107	8.908	11.129
WAC	0	45	4.179	6.149
NMI	0	115	11.247	18.219
MNO	0	30	0.678	1.756
DIT	2	8	3.087	1.21
NOC	0	60	0.571	3.401
CBO	2	56	10.363	8.403
RFC	0	306	33.678	41.693
QUALITY	34	100	85.028	14.24

Linear Regression Model-6

QUALITY = (0.065) PIM +(-0.5147) WMC + (-0.4857)WAC + (-0.2909)NMI + (-0.4916)MNO + (-1.9007)DIT + (-0.39)NOC + (-0.5369) CBO + (0.0298)RFC + 105.0304 (8)

Correlation coefficient	0.9394
Mean absolute error	3.6305
Root mean squared error	4.8894
Relative absolute error	31.1707 %
Root relative squared error	34.2482 %
Total Number of Instances	575

From the regression equations 1 to 8 and tables 1 to 6, inference can be made that software quality and various modules are inter-dependent. To validate it, following steps analysis is performed as described in subsections A and B. Subsections C and D describe the prioritization process and its validation respectively.

A. Analysis of regression model

From regression equations 1 to 8 it can be seen different versions have almost similar regression equations and also the metrics maintain corresponding impacts. Also the value of correlation coefficient indicates strong association of the metrics with quality of software. It is observed that coupling metrics DIT, CBO and NOC contribute negatively in the quality of the module and therefore selected as attributes of interest for this study. To validate the results hypothesis testing is performed.

B. Hypothesis testing

Since the proposed work is based on the assumptions that coupling metrics affect the quality of the class so hypothesis testing is performed to validate it. To conduct hypothesis testing we composed the following NULL and alternate hypothesis.

i) Null Hypothesis (H0): There is no impact of object oriented coupling on quality of the software module.

ii) Alternate Hypothesis (H_a): Object oriented coupling has negative effect on the quality of the software module.

To evaluate the Null hypothesis we performed p-value test on all six versions of the software. We obtained value of t-stat = 35.5186 and p-value as 6.48E-147. With p-value <=0.05, the null hypothesis is rejected.

Table 7. Pairwise –Correlation Analysis of metrics

Version	r (DIT, Quality)	r (NOC, Quality)	r (CBO, Quality)	Null Hypo-thesis
1.6.0	-0.54503	-0.10197	-0.77292	rejected
1.6.1	-0.53711	-0.11226	-0.76455	rejected
1.6.2	-0.54078	-0.11476	-0.75207	rejected
1.6.3	-0.5326	-0.10735	-0.7593	rejected
1.6.4	-0.53239	-0.10723	-0.75953	rejected
1.6.5	-0.53247	-0.10718	-0.75992	rejected

To evaluate the alternate hypothesis, Pearson's correlation coefficient (r) is computed. Pearson's coefficient signifies both the direction of correlation and strength of the correlation of dependent and independent variables. The pair wise value of (r) for DIT, NOC and CBO with Quality of the class is shown in table(x). We calculated the r-values for all six versions and it is found that

i. Quality of the Class is correlated with DIT, NOC and CBO measures. On the basis of results obtained the null hypothesis is therefore rejected and concluded that metrics have impact on the quality of the software.
ii. Also the correlation coefficients of all tested pairs come out to be negative, which indicates that these metrics have negative impact on the quality of the software.
iii. Also CBO is most negatively correlated with the quality of the class and hence it has the worse impact on the quality of the class.

It is also observed from the table above that CBO is most negatively affecting metric to the quality of the software. DIT is second most affecting metric and that too affect it negatively. NOC is least affecting metrics as per the results.

C. Ordering the Test Cases as per coupling measures

As inferred from the table (X) above, CBO,NOC and DIT have negative impact on the quality of the software. Also a low quality class is more susceptible to faults. So in order to prioritize the test cases ,classes may be ranked on the basis of the measures of the CBO, NOC and DIT. For analysis purpose we choose the top ten classes with maximum value of CBO. Classes where the CBO is samevalue of DIT is used to provide the ranks and where both CBO and DIT are same , value of NOC is used for ranking.

Table 8. Ordering of Test cases using coupling weights

Test Case	NAME	DIT	NOC	CBO	Rank
T1	IntrospectionHelper	2	0	57	1
T2	Project	2	0	56	2
T3	Execute	2	0	46	3
T4	Jar	6	2	44	4
T5	Main	2	0	43	5
T6	Javadoc	4	0	42	6
T7	Zip	5	1	40	7
T8	AntClassLoader	2	1	39	8
T9	FileUtils	2	0	39	9
T10	Redirector	2	0	38	10

D. Validation of Results

To validate the proposed approach, change analysis on two versions viz. 16.0 and 1.6.5 was performed. To detect the modified classes the jar files of two versions were compared with the help of Jarcomp [44] tool. Highly modified classes were identified and mapped with the top ranked classes as per the proposed strategy. It was observed that nearly 70% of the highly modified classes were correctly mapped to corresponding ranks as per the CBO, DIT and NOC weights.

V. CONCLUSION AND FUTURE SCOPE

This study proposed coupling based test case prioritization approach for object oriented testing. Proposed work analyses the coupling measures in context of regression testing and establishes the correlation of coupling with the quality of classes of the SUT. Linear regression analysis and correlation analysis were used to build the mathematical model for quality prediction of classes. A hypothesis testing was performed to validate the correlation results. Classes to test are ranked on the basis of the severity of the metrics. The work may be extended for the other metrics as well. The impact of cohesion on quality may also be explored in future investigations.

REFERENCES

[1] Gupta, Nirmal Kumar, and Mukesh Kumar Rohil. "Object Oriented Software Maintenance in Presence of Indirect Coupling." International Conference on Contemporary Computing. Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-32129-0_44

[2] Briand, Lionel C., Jurgen Wust, and Hakim Lounis. "Using coupling measurement for impact analysis in object-oriented systems." Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on. IEEE, 1999

[3] Darcy, David P., et al. "The structural complexity of software an experimental test." IEEE Transactions on software engineering 31.11 (2005): 982-995.

[4] Offutt, A. Jefferson. "Investigations of the software testing coupling effect."ACM Transactions on Software Engineering and Methodology (TOSEM)1.1 (1992): 5-20.

[5] Dalal, Siddhartha R., et al. "Model-based testing in practice." Proceedings of the 21st international conference on Software engineering. ACM, 1999.

[6] Bogdan Korel, Luay H. Tahat, and Mark Harman. 2005. Test Prioritization Using System Models. In Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM '05). IEEE Computer Society,Washington, DC,USA, 559–568.

[7] Bogdan Korel, George Koutsogiannakis, Luay H. Tahat. 2008. Application of System Models in Regression Test Suite Prioritization. In Proceedings of the IEEE International Conference on Software Maintenance (ICSM), 247–256.

[8] Tahat, Luay, et al. "Regression test suite prioritization using system models."Software Testing, Vefication and Reliability 22.7 (2012): 481-506.

[9] Chhabi Rani Panigrahi,Rajib Mall. Model-Based Regression Test Case Prioritization. ACM SIGSOFT Software Engineering Notes.Volume 35 Issue 6. 2010. 1-7.

[10] B. Korel and G. Koutsogiannakis, "Experimental Comparison of Code-Based and Model-Based Test Prioritization", In Proceedings of IEEE International Conference of Software Testing Verification and Validation Workshops, 2009.

[11] Gantait. A. Test case Generation and Prioritization from UML Model. In Proceedings of the 2011 Second International Conference on Emerging Applications of Information Technology. IEEE Computer Society Washington, DC, USA. 2011. 345-50

[12] Kundu, D., Sarma, M., Samanta, D. and Mall, R. (2009), System testing for object-oriented systems with test case prioritization. Softw. Test. Verif. Reliab., 19: 297–333. doi: 10.1002/stvr.407.

[13] Jutarat Jaroenpiboonkit and Taratip Suwannasart. Finding a Test Order using Object-Oriented Slicing Technique. In 14th Asia-Pacific Software Engineering Conference 2007. Aichi.49-56

[14] Acharya, Arup Abhinna, Prateeva Mahali, and Durga Prasad Mohapatra. "Model Based Test Case Prioritization Using Association Rule Mining." Computational Intelligence in Data Mining-Volume 3. Springer India, 2015. 429-440.

[15] Vedpal, Naresh Chauhan, Harish Kumar. A Hierarchical Test Case Prioritization Technique for Object Oriented Software. International Conference on Contemporary Computing and Informatics, Mysore,India,2014. 249-254.

[16] A. Yadav and R. A. Khan. 2009. Measuring design complexity: an inherited method perspective.SIGSOFT Softw. Eng. Notes 34, 4 (July 2009), 1-5. DOI=http://dx.doi.org/10.1145/1543405.1564532

[17] Nasib S. Gill and Sunil Sikka. 2010. New complexity model for classes in object oriented system.SIGSOFT Softw. Eng. Notes 35, 5 (October 2010), 1-7. DOI=http://dx.doi.org/10.1145/1838687.1838704

[18] Chidamber, Shyam R., and Chris F. Kemerer. Towards a metrics suite for object oriented design. Vol. 26. No. 11. ACM, 1991

[19] Tang, Mei-Huei, Ming-Hung Kao, and Mei-Hwa Chen. "An empirical study on object-oriented metrics." Software Metrics Symposium, 1999. Proceedings. Sixth International. IEEE, 1999

[20] Brian Henderson-Sellers. 1995. Object-Oriented Metrics: Measures of Complexity. Prentice-Hall, Inc., Upper Saddle River, NJ, USA

[21] Li, Wei, and Sallie Henry. "Object-oriented metrics that predict maintainability." Journal of systems and software 23.2 (1993): 111-122

[22] Li, Wei. "Another metric suite for object-oriented programming." Journal of Systems and Software 44.2 (1998): 155-162

[23] Abreu FB, Carapuça R. Object-oriented software engineering: Measuring and controlling the development process. In Proceedings of the 4th international conference on software quality 1994 Oct 3 (Vol. 186, pp. 1-8)

[24] Bansiya J, Davis CG. A hierarchical model for object-oriented design quality assessment. IEEE Transactions on software engineering. 2002 Jan;28(1):4-17

[25] Singh, Ajmer, Rajesh Bhatia, and Anita Sighrova. "Taxonomy of machine learning algorithms in software fault prediction using object oriented metrics." Procedia Computer Science132 (2018): 993-1001.

[26] Poshyvanyk, Denys, and Andrian Marcus. "The Conceptual Coupling Metrics for Object-Oriented Systems." ICSM. Vol. 6. 2006.

[27] Basili, Victor R., Lionel C. Briand, and Walcélio L. Melo. "A validation of object-oriented design metrics as quality indicators." IEEE Transactions on software engineering 22.10 (1996): 751-761

[28] Briand LC, Wüst J, Daly JW, Victor Porter D. Exploring the relationships between design measures and software quality in object-oriented systems. J Syst Softw. 2000;51(3):245-273. doi:10.1016/S0164-1212(99)00102-8.

[29] El Emam K, Melo W, Machado JC. The prediction of faulty classes using object-oriented design metrics. J Syst Softw. 2001;56:63-75. doi:10.1016/S0164-1212(00)00086-8.3

[30] Gyimothy T, Ferenc R, Siket I. Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Trans Softw Eng. 2005;31(10):897-910. doi:10.1109/TSE.2005.112.

[31] Zhou, Yuming, and Hareton Leung. "Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults." IEEE Transactions on Software Engineering 32.10 (2006): 771–789.

[32] Catal,Cagatay, and Banu Diri. "Software fault prediction with object-oriented metrics based artificial immune recognition system." Product-focused software process improvement (2007): 300-314.

[33] Alan, Oral, and Cagatay Catal. "An Outlier Detection Algorithm Based on Object-Oriented Metrics Thresholds." 2009 24th International Symposium on Computer and Information Sciences, ISCIS 2009. N.p., 2009. 567–570

[34] Singh, Yogesh, Arvinder Kaur, and Ruchika Malhotra. "Empirical Validation of Object-Oriented Metrics for Predicting Fault Proneness Models." Software Quality Journal 18.1 (2009): 3–35.

[35] Malhotra, Ruchika, and Megha Khanna. "Mining the Impact of Object Oriented Metrics for Change Prediction Using Machine Learning and Search-Based Techniques." 2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015. N.p., 2015. 228–234.

[36] Xu, Jie, Danny Ho, and Luiz Fernando Capretz. "An Empirical Validation of Object-Oriented Design Metrics for Fault Prediction." Journal of Computer Science 4.7 (2008): 583–589.

[37] Olague, Hector M. et al. "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes." IEEE Transactions on Software Engineering 33.6 (2007): 402–419.

[38] Elish, Mahmoud O., Ali H. Al-Yafei, and Muhammed Al-Mulhem. "Empirical Comparison of Three Metrics Suites for Fault Prediction in Packages of Object-Oriented Systems: A Case Study of Eclipse." Advances in Engineering Software 42.10 (2011): 852–859.

[39] Ajmer Singh, Neha Tanwar, "A Support Vector Machine based approach for effective Fault Localization" In International Conference on Soft Computing: Theory and Applications, SoCTA 2018, Jalander, India In Press

[40] Spinellis, D.: ckjm: a tool for calculating Chidamber and Kemerer Java metrics: Technical report, Athens University of Economics and Business, Athens, Greece (2006)

[41] Java Measurement Tool : jmt.stage.tigris.org

[42] STAN: Structural Analysis for JAVA : http://stan4j.com/

[43] JarcompTOOL: https://activityworkshop.net/software/jarcomp/index.html

[44] Bhandari, Parul, and Ajmer Singh. "Review of object-oriented coupling based test case selection in model based testing." In Intelligent Computing and Control Systems (ICICCS), 2017 International Conference on, pp. 1161-1165. IEEE, 2017.

Authors Profile

Mr. Ajmer Singh pursed Bachelors and Masters of Technology from Kurukshetra University, india. He is currently pursuing Ph.D. and currently working as Assistant Professor in Department of Computer Science and Engineering, DCRUST Murthal, India..He is a member of IAENG. He has 10 years of teaching experience and 4 years of Research Experience.

Dr. Rajesh Kumar Bhatia is a senior member of CSI, and he is also senior member of ACEEE, He is working as professor in Computer Science and Engineering Department, PEC University, Chandigarh. His main research work focuses on Software Testing, Software Engineering and Software Clones Detection. He has more than 20 years of teaching experience and 12 years of Research Experience.

Dr. Anita Singhrova. is Professor, Dean Faculty of Information Technology and Computer Science at Deenbandhu Chottu Ram University of Science and Technology, Murthal, Sonepat, India. She holds a Ph.D degree from GGS Indraprastha University, Delhi, India. She has completed M.E (Computer Science & Engg.) from Punjab Engineering College, Chandigarh, India and B.Tech (Computer Science) from T.I.T&S, Bhiwani, India. She has also been certified as a Java Programmer by Sun Microsystems. She possesses around twenty years of teaching experience. Her research interests include network security, mobile computing and heterogeneous networks. She has contributed in various research papers and articles published in various national and international journals and conferences.